J 5

```
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
```

```
**FILE**ID**LEXICAL
```

```
LL          EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC    AAAAAA    LL
LL          EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC    AAAAAA    LL
LL          EE          XX      XX      II      CC          AA      AA  LL
LL          EE          XX      XX      II      CC          AA      AA  LL
LL          EE            XX  XX        II      CC          AA      AA  LL
LL          EE            XX  XX        II      CC          AA      AA  LL
LL          EEEEEEEE        XX          II      CC          AA      AA  LL
LL          EEEEEEEE        XX          II      CC          AA      AA  LL
LL          EE            XX  XX        II      CC          AAAAAAAAAA  LL
LL          EE            XX  XX        II      CC          AAAAAAAAAA  LL
LL          EE          XX      XX      II      CC          AA      AA  LL              ....
LL          EE          XX      XX      II      CC          AA      AA  LL              ....
LLLLLLLLLL  EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC  AA      AA  LLLLLLLLLL      ....
LLLLLLLLLL  EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC  AA      AA  LLLLLLLLLL      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE lexical          (IDENT='V04-000',
0                          OPTLEVEL=3, ZIP,
0                          ADDRESSING_MODE(EXTERNAL=GENERAL))
1 = BEGIN
1
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 ! Facility:      Command Definition Utility, Lexical Analysis
1 !
1 ! Abstract:      This module provides the lexical analysis routines for the
1 !                Command Definition Utility.  These routines handle the
1 !                reading of CLD input files and the lexical analysis of
1 !                the files.
1 !
1 !                See the PARSE1 module for an overview of CDU parsing.
1 !
1 ! Environment:   Standard CDU environment.
1 !
1 ! Author:        Paul C. Anagnostopoulos
1 ! Creation:      29 November 1982
1 !
1 ! Modifications:
1 !
1 !       V04-006 BLS0348                 Benn Schreiber          29-AUG-1984
1 !               Put status from find_file into fab sts field.
1 !
1 !       V04-005 BLS0276                 Benn Schreiber          25-FEB-1984
1 !               Correct small problem in error reporting
1 !
1 !       V04-004 BLS0270                 Benn Schreiber           9-FEB-1984
1 !               Correct comment handling with unquoted strings
1 !
1 !       V04-003 BLS0269                 Benn Schreiber           6-FEB-1984
1 !               Convert to using LIB$FIND_FILE
1 !
```

```
1 !       V04-002 BLS0247                Benn Schreiber           28-Nov-1983
1 !               Correct obscure file opening problems.
1 !
1 !       V04-001 PCA1025                Paul C. Anagnostopoulos 25-Jul-1983
1 !               Change character class table to conform to the DEC
1 !               international character set.
1 !--
1
1
1 library 'sys$library:lib';
1 require 'cdureq';
```

```
;    70          0482  1 !       T A B L E   O F   C O N T E N T S
;    71          0483  1 !       ---------   ---   ---------------
;    72          0484  1
;    73          0485  1 forward routine
;    74          0486  1         cdu$open_next_cld,
;    75          0487  1         cdu$report_listing_heading2: novalue,
;    76          0488  1         cdu$get_next_token: novalue,
;    77          0489  1         cdu$token_must_be: novalue,
;    78          0490  1         cdu$report_syntax_error: novalue;
;    79          0491  1
;    80          0492  1 !       E X T E R N A L   R E F E R E N C E S
;    81          0493  1 !       ---------------   -------------------
;    82          0494  1
;    83          0495  1 external routine
;    84          0496  1         cdu$eject_listing_page,
;    85          0497  1         cdu$report_listing_line,
;    86          0498  1         cdu$report_rms_error,
;    87          0499  1         cli$get_value,
;    88          0500  1         lib$find_file,
;    89          0501  1         lib$signal,
;    90          0502  1         str$upcase;
;    91          0503  1
;    92        P 0504  1 $shr_msgdef(cdu,17,local,
;    93        P 0505  1         (closein,severe),
;    94        P 0506  1         (openin,error),
;    95        P 0507  1         (readerr,severe)
;    96          0508  1         );
```

```
   1 !      I N P U T   F I L E   C O N T R O L   B L O C K S
   1 !      ---------   -------   -------------   -----------
   1
   1 ! The following items define the RMS control blocks needed to open and
   1 ! read CLD input files.
   1
   1 own
   1        cld_xabdat: $xabdat(),
   1
   1        cld_esa1: block[nam$c_maxrss,byte],
   1        cld_rsa1: block[nam$c_maxrss,byte],
 P 1        cld_nam1: $nam(
 P 1                        esa=cld_esa1,
 P 1                        ess=%allocation(cld_esa1),
 P 1                        rsa=cld_rsa1,
 P 1                        rss=%allocation(cld_rsa1)
   1                        ),
   1
   1        cld_spec: $bblock[dsc$c_s_bln] preset([dsc$b_class] = dsc$k_class_d,
   1                                              [dsc$b_dtype] = dsc$k_dtype_t),
   1        out_spec: $bblock[dsc$c_s_bln] preset([dsc$b_class] = dsc$k_class_d,
   1                                              [dsc$b_dtype] = dsc$k_dtype_t),
 P 1        cld_fab: $fab(
 P 1                        fac=get,
 P 1                        fop=<sqo>,
 P 1                        nam=cld_nam1,
 P 1                        shr=get,
 P 1                        xab=cld_xabdat
   1                        ),
   1
   1        cld_buffer: block[tkn_k_max_length,byte],
 P 1        cld_rab: $rab(
 P 1                        fab=cld_fab,
 P 1                        rac=seq,
 P 1                        rop=<rah,loc,nlk>,
 P 1                        ubf=cld_buffer,
 P 1                        usz=%allocation(cld_buffer)
   1                        );
   1
   1 !      S C A N N I N G   C O N T R O L
   1 !      ---------------   -------------
   1
   1 ! The following global item counts lines as we read them from the CLD file.
   1
   1 global
   1        cdu$gl_line_number: long;
   1
   1 ! The following two items describe the token after it has been extracted
   1 ! from the CLD file.  Each token has an associated class, plus we save the
   1 ! token itself.
   1
   1 global
   1        cdu$gl_token_class: long,
   1        dbuffer(cdu$gq_token,tkn_k_max_length);
   1
   1 ! The following item keeps track of the number of errors encountered in a
   1 ! CLD file.
```

```

global
        cdu$gl_cld_errors: long;

own
! The following item tells us whether or not we are currently recovering
! from a syntax error.

        recovering: boolean,
        find_context;                   !FIND_FILE context
```

```
1 !++
1 ! Description:  This routine is called to open the next CLD input file,
1 !               which contains the definitions for one or more DCL commands.
1 !
1 ! Parameters:   none
1 !
1 ! Returns:      By reference, the FAB for the CLD input file,
1 !               or zero if no more files.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$open_next_cld
2 = BEGIN
2
2 local
2         status: long;
2
2
2 ! Determine if we have just finished with a CLD input file.
2
3 if .cld_fab[fab$w_ifi] eqlu 0 then (
3
3         ! Nope, this must be the first call, or we just recursed needing
3         ! another CLD spec.  Get the next input CLD spec.
3
3         status = cli$get_value(dtext('CLD_SPEC'),cld_spec);
3         if not .status then
3                 return 0;
3
3 ) else (
3
3         ! We just finished processing a CLD input file, so close it.
3
3         status = $close(fab=cld_fab);
3         if not .status then
3                 cdu$report_rms_error(msg(cdu$_closein),cld_fab);
2 );
2
2 ! OK, now we go into a loop in hopes of determining a file that matches the
2 ! current spec and opening it.
2
3 loop (
3         local rms_stv;
3
3         status = lib$find_file(cld_spec,out_spec,find_context,
3                                $descriptor('.CLD'),0,rms_stv,%REF(2));
3         cld_fab[fab$b_fns] = .out_spec[dsc$w_length];
3         cld_fab[fab$l_fna] = .out_spec[dsc$a_pointer];
3         cld_fab[fab$l_sts] = .status;
3         cld_fab[fab$l_stv] = .rms_stv;
3         if .status eqlu rms$_nmf then exitloop;
3
3         ! If we have a file to open, then do it.  Otherwise report the error
3         ! and loop for another try.
3
4         if .status then (
4                       status = $open(fab=cld_fab);
5                       if .status then (
5                               status = $connect(rab=cld_rab);
6                               if .status then (
6                                       cdu$gl_line_number = 0;
6                                       return cld_fab;
5                               ) else
5                                       cdu$report_rms_error(msg(cdu$_openin),cld_rab);
4                       ) else
4                               cdu$report_rms_error(msg(cdu$_openin),cld_fab);
3               ) else
3                       cdu$report_rms_error(msg(cdu$_openin),.find_context);
2 );
2
2 ! We don't have any more files that match the spec.  Recurse to get the
2 ! next spec.
2
2 return cdu$open_next_cld();
2
1 END;


                                                        .TITLE  LEXICAL
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

          43  45  50  53  5F  44  4C  43  00000 P.AAB:  .ASCII  \CLD_SPEC\                        ;
                             010E0008  00008 P.AAA:  .LONG   17694728                          ;
                            00000000'  0000C         .ADDRESS P.AAB                            ;
                          44  4C  43  2E  00010 P.AAD:  .ASCII  \.CLD\                            ;
                             00000004  00014 P.AAC:  .LONG   4                                 ;
                            00000000'  00018         .ADDRESS P.AAD                            ;

                                                        .PSECT  $OWN$,NOEXE,2

                                   12  00000 CLD_XABDAT:
                                                        .BYTE   18                                ;
                                   2C  00001            .BYTE   44                                ;
                                 0000  00002            .WORD   0                                 ;
                             00000000  00004            .LONG   0                                 ;
                                 0000  00008            .WORD   0                                 ;
                                 0000  0000A            .WORD   0                                 ;
                            00000000#  0000C            .LONG   0[2]                              ;
                            00000000#  00014            .LONG   0[2]                              ;
                             00000000  0001C            .LONG   0                                 ;
                             00000000  00020            .LONG   0                                 ;
                            00000000#  00024            .LONG   0[2]                              ;
                                       0002C CLD_ESA1:
                                                        .BLKB   255
                                       0012B            .BLKB   1
                                       0012C CLD_RSA1:
                                                        .BLKB   255
                                       0022B            .BLKB   1
                                   02  0022C CLD_NAM1:
                                                        .BYTE   2                                 ;
                                   60  0022D            .BYTE   96                                ;
```

```
       FF  0022E            .BYTE    -1                          ;
       00  0022F            .BYTE    0                           ;
00000000'  00230            .ADDRESS CLD_RSA1                    ;
       00  00234            .BYTE    0                           ;
       00  00235            .BYTE    0                           ;
       FF  00236            .BYTE    -1                          ;
       00  00237            .BYTE    0                           ;
00000000'  00238            .ADDRESS CLD_ESA1                    ;
00000000   0023C            .LONG    0                           ;
    0000#  00240            .WORD    0[8]                        ;
    0000#  00250            .WORD    0[3]                        ;
    0000#  00256            .WORD    0[3]                        ;
00000000   0025C            .LONG    0                           ;
00000000   00260            .LONG    0                           ;
       00  00264            .BYTE    0                           ;
       00  00265            .BYTE    0                           ;
       00  00266            .BYTE    0                           ;
       00  00267            .BYTE    0                           ;
       00  00268            .BYTE    0                           ;
       00  00269            .BYTE    0                           ;
      00#  0026A            .BYTE    0[2]                        ;
00000000   0026C            .LONG    0                           ;
00000000   00270            .LONG    0                           ;
00000000   00274            .LONG    0                           ;
00000000   00278            .LONG    0                           ;
00000000   0027C            .LONG    0                           ;
00000000   00280            .LONG    0                           ;
00000000#  00284            .LONG    0[2]                        ;
      00#  0028C CLD_SPEC:
                            .BYTE    0[2]                        ;
    02  0E 0028E            .BYTE    14, 2                       ;
           00290            .BLKB    4
      00#  00294 OUT_SPEC:
                            .BYTE    0[2]                        ;
    02  0E 00296            .BYTE    14, 2                       ;
           00298            .BLKB    4
       03  0029C CLD_FAB:.BYTE    3                              ;
       50  0029D            .BYTE    80                          ;
     0000  0029E            .WORD    0                           ;
00000040   002A0            .LONG    64                          ;
00000000   002A4            .LONG    0                           ;
00000000   002A8            .LONG    0                           ;
00000000   002AC            .LONG    0                           ;
     0000  002B0            .WORD    0                           ;
       02  002B2            .BYTE    2                           ;
       02  002B3            .BYTE    2                           ;
00000000   002B4            .LONG    0                           ;
       00  002B8            .BYTE    0                           ;
       00  002B9            .BYTE    0                           ;
       00  002BA            .BYTE    0                           ;
       02  002BB            .BYTE    2                           ;
00000000   002BC            .LONG    0                           ;
00000000'  002C0            .ADDRESS CLD_XABDAT                  ;
00000000'  002C4            .ADDRESS CLD_NAM1                    ;
00000000   002C8            .LONG    0                           ;
00000000   002CC            .LONG    0                           ;
       00  002D0            .BYTE    0                           ;
      00  002D1             .BYTE    0                                                    ;
    0000  002D2             .WORD    0                                                    ;
00000000  002D4             .LONG    0                                                    ;
    0000  002D8             .WORD    0                                                    ;
      00  002DA             .BYTE    0                                                    ;
      00  002DB             .BYTE    0                                                    ;
00000000  002DC             .LONG    0                                                    ;
00000000  002E0             .LONG    0                                                    ;
    0000  002E4             .WORD    0                                                    ;
      00  002E6             .BYTE    0                                                    ;
      00  002E7             .BYTE    0                                                    ;
00000000  002E8             .LONG    0                                                    ;
          002EC CLD_BUFFER:
                            .BLKB    255
          003EB             .BLKB    1
      01  003EC CLD_RAB:    .BYTE    1                                                    ;
      44  003ED             .BYTE    68                                                   ;
    0000  003EE             .WORD    0                                                    ;
00110200  003F0             .LONG    1114624                                              ;
00000000  003F4             .LONG    0                                                    ;
00000000  003F8             .LONG    0                                                    ;
   0000#  003FC             .WORD    0[3]                                                 ;
    0000  00402             .WORD    0                                                    ;
00000000  00404             .LONG    0                                                    ;
    0000  00408             .WORD    0                                                    ;
      00  0040A             .BYTE    0                                                    ;
      00  0040B             .BYTE    0                                                    ;
    00FF  0040C             .WORD    255                                                  ;
    0000  0040E             .WORD    0                                                    ;
00000000' 00410             .ADDRESS CLD_BUFFER                                           ;
00000000  00414             .LONG    0                                                    ;
00000000  00418             .LONG    0                                                    ;
00000000  0041C             .LONG    0                                                    ;
      00  00420             .BYTE    0                                                    ;
      00  00421             .BYTE    0                                                    ;
      00  00422             .BYTE    0                                                    ;
      00  00423             .BYTE    0                                                    ;
00000000  00424             .LONG    0                                                    ;
00000000' 00428             .ADDRESS CLD_FAB                                              ;
00000000  0042C             .LONG    0                                                    ;
          00430 RECOVERING:
                            .BLKB    1
          00431             .BLKB    3
          00434 FIND_CONTEXT:
                            .BLKB    4

                            .PSECT  $GLOBAL$,NOEXE,2

          00000 CDU$GL_LINE_NUMBER::
                            .BLKB    4
          00004 CDU$GL_TOKEN_CLASS::
                            .BLKB    4
    00FF  00008 CDU$GQ_TOKEN::
                            .WORD    255                                                  ;
   00 00  0000A             .BYTE    0, 0                                                 ;
00000000' 0000C             .ADDRESS CDU$GQ_TOKEN+8                                       ;
          00010             .BLKB    255
```

```
                                        0010F            .BLKB   1
                                        00110 CDU$GL_CLD_ERRORS::
                                                         .BLKB   4

                                                         .EXTRN  CDU$EJECT_LISTING_PAGE
                                                         .EXTRN  CDU$REPORT_LISTING_LINE
                                                         .EXTRN  CDU$REPORT_RMS_ERROR
                                                         .EXTRN  CLI$GET_VALUE, LIB$FIND_FILE
                                                         .EXTRN  LIB$SIGNAL, STR$UPCASE
                                                         .EXTRN  SYS$CLOSE, SYS$OPEN
                                                         .EXTRN  SYS$CONNECT

                                                         .PSECT  $CODE$,NOWRT,2

                                  0004  00000            .ENTRY  CDU$OPEN_NEXT_CLD, Save R2                          : 0588
               5E             08   C2   00002            SUBL2   #8, SP
                     0000'    CF   B5   00005            TSTW    CLD_FAB+2                                           : 0597
                              18   12   00009            BNEQ    1$
                     0000'    CF   9F   0000B            PUSHAB  CLD_SPEC                                            : 0602
                     0000'    CF   9F   0000F            PUSHAB  P.AAA
    00000000G  00             02   FB   00013            CALLS   #2, CLI$GET_VALUE
               52             50   D0   0001A            MOVL    R0, STATUS
               25             52   E8   0001D            BLBS    STATUS, 3$                                          : 0603
                            00B6   31   00020            BRW     9$                                                  : 0604
                     0000'    CF   9F   00023  1$:       PUSHAB  CLD_FAB                                             : 0610
    00000000G  00             01   FB   00027            CALLS   #1, SYS$CLOSE
               52             50   D0   0002E            MOVL    R0, STATUS
               11             52   E8   00031            BLBS    STATUS, 3$                                          : 0611
                     0000'    CF   9F   00034            PUSHAB  CLD_FAB                                             : 0612
                 00111054     8F   DD   00038            PUSHL   #1118292
    00000000G  00             02   FB   0003E  2$:       CALLS   #2, CDU$REPORT_RMS_ERROR
               6E             02   D0   00045  3$:       MOVL    #2, (SP)                                            : 0622
                              5E   DD   00048            PUSHL   SP
                       08     AE   9F   0004A            PUSHAB  RMS_STV                                             : 0621
                              7E   D4   0004D            CLRL    -(SP)
                     0000'    CF   9F   0004F            PUSHAB  P.AAC                                               : 0622
                     0000'    CF   9F   00053            PUSHAB  FIND_CONTEXT                                        : 0621
                     0000'    CF   9F   00057            PUSHAB  OUT_SPEC
                     0000'    CF   9F   0005B            PUSHAB  CLD_SPEC
    00000000G  00             07   FB   0005F            CALLS   #7, LIB$FIND_FILE
               52             50   D0   00066            MOVL    R0, STATUS
        0000'  CF    0000'    CF   90   00069            MOVB    OUT_SPEC, CLD_FAB+52                                : 0623
        0000'  CF    0000'    CF   D0   00070            MOVL    OUT_SPEC+4, CLD_FAB+44                              : 0624
        0000'  CF                  52   D0   00077       MOVL    STATUS, CLD_FAB+8                                   : 0625
        0000'  CF          04 AE   D0   0007C            MOVL    RMS_STV, CLD_FAB+12                                 : 0626
    000182CA   8F             52   D1   00082            CMPL    STATUS, #99018                                      : 0627
                              48   13   00089            BEQL    8$
               38             52   E9   0008B            BLBC    STATUS, 6$                                          : 0632
                     0000'    CF   9F   0008E            PUSHAB  CLD_FAB                                             : 0633
    00000000G  00             01   FB   00092            CALLS   #1, SYS$OPEN
               52             50   D0   00099            MOVL    R0, STATUS
               21             52   E9   0009C            BLBC    STATUS, 5$                                          : 0634
                     0000'    CF   9F   0009F            PUSHAB  CLD_RAB                                             : 0635
    00000000G  00             01   FB   000A3            CALLS   #1, SYS$CONNECT
               52             50   D0   000AA            MOVL    R0, STATUS
               0A             52   E9   000AD            BLBC    STATUS, 4$                                          : 0636
                     0000'    CF   D4   000B0            CLRL    CDU$GL_LINE_NUMBER                                  : 0637
```

```
                                  50      0000'  CF   9E 000B4            MOVAB   CLD_FAB, R0                            ; 0638
                                                      04 000B9            RET                                            ;
                                          0000'  CF   9F 000BA 4$:        PUSHAB  CLD_RAB                                ; 0640
                                                 0A   11 000BE            BRB     7$                                     ;
                                          0000'  CF   9F 000C0 5$:        PUSHAB  CLD_FAB                                ; 0642
                                                 04   11 000C4            BRB     7$                                     ;
                                          0000'  CF   DD 000C6 6$:        PUSHL   FIND_CONTEXT                           ; 0644
                                      0011109A   8F   DD 000CA 7$:        PUSHL   #1118362                               ;
                                              FF6B    31 000D0            BRW     2$                                     ;
                           FF28   CF             00   FB 000D3 8$:        CALLS   #0, CDU$OPEN_NEXT_CLD                  ; 0650
                                                      04 000D8            RET                                            ;
                                                 50   D4 000D9 9$:        CLRL    R0                                     ; 0652
                                                      04 000DB            RET                                            ;

; Routine Size:  220 bytes,    Routine Base:  $CODE$ + 0000
;   244          0653   1 !++
;   245          0654   1 ! Description:  This routine is called from the LISTING module to generate
;   246          0655   1 !               the second heading line for a page header.  This line
;   247          0656   1 !               contains the CLD file spec and its creation date.
;   248          0657   1 !
;   249          0658   1 ! Parameters:   None.
;   250          0659   1 !
;   251          0660   1 ! Returns:      Nothing.
;   252          0661   1 !
;   253          0662   1 ! Notes:
;   254          0663   1 !--
;   255          0664   1
;   256          0665   1 GLOBAL ROUTINE cdu$report_listing_heading2      : novalue
;   257          0666   2 = BEGIN
;   258          0667   2
;   259          0668   2 bind
;   260          0669   2         nam = .cld_fab[fab$l_nam]: block[.byte];
;   261          0670   2
;   262          0671   2
;   263          0672   2 ! Generate a heading line with the CLD file's revision date, spec, and
;   264          0673   2 ! revision number.
;   265          0674   2
;   266          0675   2 cdu$report_listing_line(msg(cdu$_heading2),nobabble+4,
;   267          0676   2                         cld_xabdat[xab$q_rdt],
;   268          0677   2                         .nam[nam$b_rsl],.nam[nam$l_rsa],
;   269          0678   2                         .cld_xabdat[xab$w_rvn]);
;   270          0679   2
;   271          0680   2 return;
;   272          0681   2
;   273          0682   1 END;


                                                          .EXTRN  CDU$_HEADING2

                                          0000 00000      .ENTRY  CDU$REPORT_LISTING_HEADING2, Save nothing   ; 0665
                    50      0000'   CF    D0 00002        MOVL    CLD_FAB+40, R0                              ; 0669
                    7E      0000'   CF    3C 00007        MOVZWL  CLD_XABDAT+8, -(SP)                         ; 0678
                              04    A0    DD 0000C        PUSHL   4(R0)                                       ; 0677
                    7E        03    A0    9A 0000F        MOVZBL  3(R0), -(SP)                                ;
                            0000'   CF    9F 00013        PUSHAB  CLD_XABDAT+12                               ; 0676
                         00010004   8F    DD 00017        PUSHL   #65540                                      ;
                         00000000G  8F    DD 0001D        PUSHL   #CDU$_HEADING2                              ;
              00000000G  00         06    FB 00023        CALLS   #6, CDU$REPORT_LISTING_LINE                 ;
                                          04 0002A        RET                                                 ; 0682

; Routine Size:  43 bytes,    Routine Base:  $CODE$ + 00DC
```

```
1 !++
1 ! Description:  This routine is called to obtain the next token from the
1 !               CLD file being compiled.
1 !
1 ! Parameters:   hint            Optional, by value, a hint about the fact that
1 !                               the caller expects an h-string, which is a
1 !                               quoted string or arbitrary stuff ending at
1 !                               certain delimiters.
1 !
1 ! Returns:      Nothing
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$get_next_token(hint: long): novalue
2 = BEGIN
2
2 builtin
2         nullparameter;
2
2 linkage
2         jsb_for_speed = jsb(; register=0);
2
2 own
2         line_index: long,
2         char: byte,
2         char_saved: boolean;
2
2 local
2         status: long;
```

```
!       C H A R A C T E R   C L A S S   T A B L E
!       -----------------   ---------   ---------

! The following table maps each of the 256 ASCII character codes into
! their corresponding character class.

own
        char_class: vector[256,byte] initial(byte(
                        rep 3 of (tkn_k_invalid),           ! NUL - STX
                        tkn_k_eof,                          ! ETX
                        rep 5 of (tkn_k_invalid),           ! EOT - BS
                        tkn_k_whitespace,                   ! HT
                        rep 2 of (tkn_k_invalid),           ! LF - VT
                        tkn_k_ignored,                      ! FF
                        tkn_k_eol,                          ! CR
                        rep 18 of (tkn_k_invalid),          ! SO - US
                        tkn_k_whitespace,                   ! space
                        tkn_k_comment,                      ! !
                        tkn_k_string,                       ! "
                        tkn_k_invalid,                      ! #
                        tkn_k_symbol,                       ! $
                        rep 3 of (tkn_k_invalid),           ! % & '
                        tkn_k_open_paren,                   ! (
                        tkn_k_close_paren,                  ! )
                        rep 2 of (tkn_k_invalid),           ! * +
                        tkn_k_comma,                        ! ,
                        tkn_k_invalid,                      ! -
                        tkn_k_dot,                          ! .
                        tkn_k_invalid,                      ! /
                        rep 10 of (tkn_k_symbol),           ! 0 - 9
                        rep 2 of (tkn_k_invalid),           ! : ;
                        tkn_k_open_angle,                   ! <
                        tkn_k_equal,                        ! =
                        tkn_k_close_angle,                  ! >
                        rep 2 of (tkn_k_invalid),           ! ? @
                        rep 26 of (tkn_k_symbol),           ! A - Z
                        rep 4 of (tkn_k_invalid),           ! [ \ ] ^
                        tkn_k_symbol,                       ! _
                        tkn_k_invalid,                      ! 
                        rep 26 of (tkn_k_symbol),           ! a - z
                        rep 5 of (tkn_k_invalid),           ! { | } ~ DEL
                        rep 64 of (tkn_k_invalid),          ! bit 7 on...
                        rep 63 of (tkn_k_symbol),           ! int'l alphabetics
                        rep 1 of (tkn_k_invalid)
                        ));
```

```
2 ! This internal routine is called to obtain the next character from the CLD
2 ! file.  It handles reading lines from the file and pulling characters from
2 ! the lines.  It also writes the lines into the listing file.
2
2 ROUTINE get_next_char           : jsb_for_speed
3 = BEGIN
3
3 local
3         status: long;
3
3
3 ! If the line number is zero, or we've run out of characters on the current
3 ! line, let's get another line.
3
4 if .cdu$gl_line_number eqlu 0 or .line_index gtru .cld_rab[rab$w_rsz] then (
4
4         ! Sit in a loop reading lines until we get one that isn't null.
4         ! If we get end of file, return an ETX character.  List the lines
4         ! as we go.
4
5         do (
5                 status = $get(rab=cld_rab);
5                 if .status eqlu rms$_eof then
5                         return ETX;
5                 if not .status then
5                         cdu$report_rms_error(msg(cdu$_readerr),cld_rab);
5                 increment(cdu$gl_line_number);
5                 if .cld_rab[rab$w_rsz] eqlu 1 and ch$rchar(.cld_rab[rab$l_rbf]) eqlu FF then
5                         cdu$eject_listing_page()
5                 else
5                         cdu$report_listing_line(msg(cdu$_listline),nobabble+3,
5                                 .cdu$gl_line_number,.cld_rab[rab$w_rsz],.cld_rab[rab$l_rbf]);
4         ) until .cld_rab[rab$w_rsz] nequ 0;
4
4         ! Reset the line index to zero.
4
4         line_index = 0;
3 );
3
3 ! Now we can pull a character from the line.  We always pretend that there
3 ! is a carriage return at the end.  Make sure to increment the line index.
3
4 if .line_index lssu .cld_rab[rab$w_rsz] then (
4         increment(line_index);
4         return ch$rchar(.cld_rab[rab$l_rbf]+.line_index-1);
4 ) else (
4         increment(line_index);
4         return CR;
3 );
3
2 END;


                                                          .PSECT  $OWN$,NOEXE,2

                                        00438 LINE_INDEX:
```

```
                                                                     .BLKB    4
                                         0043C CHAR:                 .BLKB    1
                                         0043D CHAR_SAVED:
                                                                     .BLKB    1
                                         0043E                       .BLKB    2
                                    00#  00440 CHAR_CLASS:
                                                                     .BYTE    0[3]                                          :
                                    04   00443                       .BYTE    4                                             :
                                    00#  00444                       .BYTE    0[5]                                          :
                                    02   00449                       .BYTE    2                                             :
                                    00#  0044A                       .BYTE    0[2]                                          :
                               03   01   0044C                       .BYTE    1, 3                                          :
                                    00#  0044E                       .BYTE    0[18]                                         :
               0D   00   0B    0A   02   00460                       .BYTE    2, 10, 11, 0, 13                              :
                                    00#  00465                       .BYTE    0[3]                                          :
                               08   07   00468                       .BYTE    7, 8                                          :
                                    00#  0046A                       .BYTE    0[2]                                          :
                    00   09    00   05   0046C                       .BYTE    5, 0, 9, 0                                    :
                                    0D#  00470                       .BYTE    13[10]                                        :
                                    00#  0047A                       .BYTE    0[2]                                          :
                         0F    06   0E   0047C                       .BYTE    14, 6, 15                                     :
                                    00#  0047F                       .BYTE    0[2]                                          :
                                    0D#  00481                       .BYTE    13[26]                                        :
                                    00#  0049B                       .BYTE    0[4]                                          :
                               00   0D   0049F                       .BYTE    13, 0                                         :
                                    0D#  004A1                       .BYTE    13[26]                                        :
                                    00#  004BB                       .BYTE    0[5]                                          :
                                    00#  004C0                       .BYTE    0[64]                                         :
                                    0D#  00500                       .BYTE    13[63]                                        :
                                    00   0053F                       .BYTE    0                                             :

                                                                     .EXTRN   SYS$GET, CDU$_LISTLINE

                                                                     .PSECT   $CODE$,NOWRT,2

                               52   DD 00000 GET_NEXT_CHAR:
                                                                     PUSHL    R2                                            ; 0762
                         0000'  CF   D5 00002                        TSTL     CDU$GL_LINE_NUMBER                            ; 0772
                                0B   13 00006                        BEQL     1$                                            :
0000'  CF   0000'  CF        10      00   ED 0000B                   CMPZV    #0, #16, CLD_RAB+34, LINE_INDEX               :
                                75   1E 00011                        BGEQU    6$                                            :
                         0000'  CF   9F 00013 1$:                    PUSHAB   CLD_RAB                                       ; 0779
          00000000G  00         01   FB 00017                        CALLS    #1, SYS$GET                                   :
                     52         50   D0 0001E                        MOVL     R0, STATUS                                    :
          0001827A   BF         52   D1 00021                        CMPL     STATUS, #98938                                ; 0780
                                05   12 00028                        BNEQ     2$                                            :
                     50         03   D0 0002A                        MOVL     #3, R0                                        ; 0781
                                7D   11 0002D                        BRB      8$                                            :
                     11         52   E8 0002F 2$:                    BLBS     STATUS, 3$                                    ; 0782
                         0000'  CF   9F 00032                        PUSHAB   CLD_RAB                                       ; 0783
                      001110B4  8F   DD 00036                        PUSHL    #1118388                                      :
          00000000G  00         02   FB 0003C                        CALLS    #2, CDU$REPORT_RMS_ERROR                      :
                         0000'  CF   D6 00043 3$:                    INCL     CDU$GL_LINE_NUMBER                            ; 0784
                     01  0000'  CF   B1 00047                        CMPW     CLD_RAB+34, #1                                ; 0785
                                10   12 0004C                        BNEQ     4$                                            :
                     0C  0000'  DF   91 0004E                        CMPB     @CLD_RAB+40, #12                              :
                                09   12 00053                        BNEQ     4$                                            :
```

```
                              00000000G  00            00  FB 00055       CALLS   #0, CDU$EJECT_LISTING_PAGE                   : 0786
                                                       20  11 0005C       BRB     5$                                           :
                                               0000'   CF  DD 0005E 4$:   PUSHL   CLD_RAB+40                                   : 0789
                                         7E    0000'   CF  3C 00062       MOVZWL  CLD_RAB+34, -(SP)                            :
                                               0000'   CF  DD 00067       PUSHL   CDU$GL_LINE_NUMBER                           :
                                           00010003    8F  DD 0006B       PUSHL   #65539                                       : 0788
                                           00000000G   8F  DD 00071       PUSHL   #CDU$_LISTLINE                               :
                              00000000G  00            05  FB 00077       CALLS   #5, CDU$REPORT_LISTING_LINE                  :
                                               0000'   CF  B5 0007E 5$:   TSTW    CLD_RAB+34                                   : 0790
                                                       8F  13 00082       BEQL    1$                                           :
                                               0000'   CF  D4 00084       CLRL    LINE_INDEX                                   : 0794
    0000'  CF      0000'  CF         10                00  ED 00088 6$:   CMPZV   #0, #16, CLD_RAB+34, LINE_INDEX              : 0800
                                                       12  1B 00091       BLEQU   7$                                           :
                                               0000'   CF  D6 00093       INCL    LINE_INDEX                                   : 0801
                         50      0000'   CF    0000'   CF  C1 00097       ADDL3   LINE_INDEX, CLD_RAB+40, R0                   : 0802
                                         50       FF   A0  9A 0009F       MOVZBL  -1(R0), R0                                   : 0803
                                                       07  11 000A3       BRB     8$                                           :
                                               0000'   CF  D6 000A5 7$:   INCL    LINE_INDEX                                   : 0804
                                         50            0D  D0 000A9       MOVL    #13, R0                                      : 0805
                                         52            8E  D0 000AC 8$:   MOVL    (SP)+, R2                                    : 0808
                                                           05 000AF       RSB                                                  :

; Routine Size:  176 bytes,    Routine Base:  $CODE$ + 0107
```

```
2 ! The following internal routine is called to get an h-string, if the
2 ! caller has told us that one is expected.  An h-string is either a
2 ! normal quoted string, or it is an arbitrary sequence of characters ending
2 ! at certain delimiters or at end of line.
2
2 ROUTINE get_h_string    : novalue
3 = BEGIN
3
3 local
3         quoted: boolean,
3         class: long;
3
3
3 ! Clear the token buffer.
3
3 cdu$gq_token[len] = 0;
3
3 ! Pull a character from the CLD file.  We may already have one saved from
3 ! the previous call.
3
3 if not .char_saved then
3         char = get_next_char();
3 char_saved = true;
3
3 ! Pass up any leading whitespace.
3
3 while .char_class[.char] eqlu tkn_k_whitespace do
3         char = get_next_char();
3
3 ! If we now have a quotation mark, then it's a quoted string.  Just return
3 ! and let the normal routine process it.
3
3 if .char_class[.char] eqlu tkn_k_string then
3         return;
3
3 ! Sit in a loop and collect the characters into the global token buffer.
3 ! We quit when we encounter one of the ending delimiters, or if we hit end
3 ! of line.
3
4 loop (
4         case .char_class[.char] from 0 to tkn_k_max_class of set
4         [tkn_k_eol,
4         tkn_k_comma,
4         tkn_k_equal,
4         tkn_k_comment,
4         tkn_k_open_paren,
4         tkn_k_close_paren]:     exitloop;
4
4         [inrange,
4         outrange]:              ;
4         tes;
4         ch$wchar(.char, .cdu$gq_token[ptr]+.cdu$gq_token[len]);
4         increment(cdu$gq_token[len]);
4         char = get_next_char();
3 );
3
3 ! Set the token globals to say it's a string.
```

```
;  461         0866  3
;  462         0867  3 cdu$gl_token_class = tkn_k_string;
;  463         0868  3
;  464         0869  3 ! Upcase the string for compatibility with the old CDU, even though that
;  465         0870  3 ! doesn't really seem reasonable.
;  466         0871  3
;  467         0872  3 str$upcase(cdu$gq_token,cdu$gq_token);
;  468         0873  3
;  469         0874  3 return;
;  470         0875  3
;  471         0876  2 END;



                                          0FFC 00000 GET_H_STRING:
                                                           .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11              ; 0814
                              0000'  CF   B4 00002         CLRW     CDU$GQ_TOKEN                                      ; 0824
                     08       0000'  CF   E8 00006         BLBS     CHAR_SAVED, 1$                                    ; 0829
                                   FF42   30 0000B         BSBW     GET_NEXT_CHAR                                     ; 0830
              0000'  CF              50   90 0000E         MOVB     R0, CHAR                                          ;
              0000'  CF              01   90 00013 1$:     MOVB     #1, CHAR_SAVED                                    ; 0831
                                     08   11 00018         BRB      3$                                                ; 0835
                                   FF33   30 0001A 2$:     BSBW     GET_NEXT_CHAR                                     ; 0836
              0000'  CF              50   90 0001D         MOVB     R0, CHAR                                          ;
                     50       0000'  CF   9A 00022 3$:     MOVZBL   CHAR, R0                                          ; 0835
                     02       0000'CF40   91 00027         CMPB     CHAR_CLASS[R0], #2                                ;
                                     EB   13 0002D         BEQL     2$                                                ;
                     50       0000'  CF   9A 0002F         MOVZBL   CHAR, R0                                          ; 0841
                     0B       0000'CF40   91 00034         CMPB     CHAR_CLASS[R0], #11                               ;
                                     5B   13 0003A         BEQL     8$                                                ;
                     50       0000'  CF   9A 0003C 4$:     MOVZBL   CHAR, R0                                          ; 0849
           0F        00       0000'CF40   8F 00041         CASEB    CHAR_CLASS[R0], #0, #15                           ;
  003B     0020      0020          0020      00048 5$:     .WORD    6$-5$,-                                           ;
  003B     003B      003B          0020      00050                  6$-5$,-                                           ;
  0020     003B      0020          003B      00058                  6$-5$,-                                           ;
  0020     0020      0020          0020      00060                  7$-5$,-                                           ;
                                                                    6$-5$,-                                           ;
                                                                    7$-5$,-                                           ;
                                                                    7$-5$,-                                           ;
                                                                    7$-5$,-                                           ;
                                                                    7$-5$,-                                           ;
                                                                    6$-5$,-                                           ;
                                                                    7$-5$,-                                           ;
                                                                    6$-5$,-                                           ;
                                                                    6$-5$,-                                           ;
                                                                    6$-5$,-                                           ;
                                                                    6$-5$,-                                           ;
                                                                    6$-5$                                             ;
                     51       0000'  CF   3C 00068 6$:     MOVZWL   CDU$GQ_TOKEN, R1                                  ; 0860
                     51       0000'  CF   C0 0006D         ADDL2    CDU$GQ_TOKEN+4, R1                                ;
                     61              50   90 00072         MOVB     R0, (R1)                                          ;
                              0000'  CF   B6 00075         INCW     CDU$GQ_TOKEN                                      ; 0861
                                   FED4   30 00079         BSBW     GET_NEXT_CHAR                                     ; 0862
              0000'  CF              50   90 0007C         MOVB     R0, CHAR                                          ;
                                     B9   11 00081         BRB      4$                                                ; 0842
```

```
                          0000'  CF            0B  D0 00083 7$:      MOVL    #11, CDU$GL_TOKEN_CLASS                         ; 0867
                                        0000'  CF  9F 00088          PUSHAB  CDU$GQ_TOKEN                                    ; 0872
                                        0000'  CF  9F 0008C          PUSHAB  CDU$GQ_TOKEN                                    ;
                       00000000G  00           02  FB 00090          CALLS   #2, STR$UPCASE                                  ;
                                                   04 00097 8$:      RET                                                     ; 0876

; Routine Size:  152 bytes,    Routine Base:  $CODE$ + 01B7
```

```
2 ! If the line number is zero, then a new CLD file has just been opened.
2 ! Reset the error counter, the error recovery flag, and the flag that
2 ! tells us that a character is being saved for processing.
2
3 if .cdu$gl_line_number eqlu 0 then (
3         cdu$gl_cld_errors = 0;
3         recovering = false;
3         char_saved = false;
2 );
2
2 ! If we have been told that the caller is expecting an h-string, then we
2 ! call a special internal routine to get it.  If we end up with a null
2 ! string, then it was either a normal quoted string, or the h-string
2 ! was null.
2
2 if not nullparameter(1) then
3         if .hint eqlu tkn_k_h_string then (
3                 get_h_string();
3                 if .cdu$gq_token[len] nequ 0 then
3                         return;
2         );
2
2 ! We cycle through the following loop once for each "noise" character,
2 ! until we finally find an interesting one.  Then we collect the token
2 ! and return.
2
3 loop (
3
3         ! Pull a character from the CLD file.  We may already have one
3         ! saved from the previous call.  Initialize the token globals
3         ! with the character.
3
3         if not .char_saved then
3                 char = get_next_char();
3         char_saved = false;
3         cdu$gq_token[len] = 1;
3         ch$wchar(.char, .cdu$gq_token[ptr]);
3
3         ! Determine the class of the character by looking it up in the
3         ! class table.  Initialize the token globals with the class.
3
3         cdu$gl_token_class = .char_class[.char];
3
3         ! Case on the character class.
3
3         case .cdu$gl_token_class from 0 to tkn_k_max_class of set
3         [tkn_k_invalid]:
3
3                 ! Invalid characters result in an error message, and then
3                 ! they are ignored.
3
3                 cdu$report_syntax_error(msg(cdu$_invchar),1,.line_index);
3
3         [tkn_k_ignored,
3          tkn_k_whitespace,
3          tkn_k_eol]:
3
```

```
;  3                 ! All these characters are just ignored.
;  3
;  3                 ;
;  3
;  3         [tkn_k_eof,
;  3          tkn_k_comma,
;  3          tkn_k_equal,
;  3          tkn_k_open_paren,
;  3          tkn_k_close_paren,
;  3          tkn_k_open_angle,
;  3          tkn_k_close_angle,
;  3          tkn_k_dot]:
;  3
;  3                 ! All of these single-character tokens are very simple.
;  3                 ! We're all done.
;  3
;  3                 return;
;  3
;  3         [tkn_k_comment]:
;  3
;  3                 ! To handle a comment, we want to ignore the rest of the
;  3                 ! line.  Advance the line index off the face of the earth
;  3                 ! so that GET_NEXT_CHAR will get the next line.
;  3
;  3                 line_index = 999999;
;  3
;  3         [tkn_k_string]:
;  3
;  3                 ! To collect a string, we keep pulling characters and
;  3                 ! adding them to the string.  If we hit end-of-line, that's
;  3                 ! an error.  If we hit two string delimiters in a row, then
;  3                 ! we include one in the string and keep going.
;  3
;  4                 (local
;  4                         char2: byte;
;  4
;  4                 cdu$gq_token[len] = 0;
;  5                 loop (
;  5                         char2 = get_next_char();
;  5                         selectoneu .char_class[.char2] of set
;  5                         [tkn_k_eol]:
;  6                                 (cdu$report_syntax_error(msg(cdu$_missquote));
;  5                                 exitloop;);
;  5
;  5                         [tkn_k_string]:
;  6                                 if (char = get_next_char()) eqlu .char2 then (
;  6                                         ch$wchar(.char2, .cdu$gq_token[ptr]+.cdu$gq_token[len]);
;  6                                         increment(cdu$gq_token[len]);
;  6                                 ) else (
;  6                                         char_saved = true;
;  6                                         exitloop;
;  5                                 );
;  5
;  5                         [otherwise]:
;  6                                 (ch$wchar(.char2, .cdu$gq_token[ptr]+.cdu$gq_token[len]);
;  5                                 increment(cdu$gq_token[len]););
;  5                         tes;
;   587          0991  4                 );
;   588          0992  4
;   589          0993  3                 return;);
;   590          0994  3
;   591          0995  3         [tkn_k_h_string]:
;   592          0996  3
;   593          0997  3                 ! There are no characters of class h-string.
;   594          0998  3
;   595          0999  3                 signal(msg(cdu$_inthchar));
;   596          1000  3
;   597          1001  3         [tkn_k_symbol]:
;   598          1002  3
;   599          1003  3
;   600          1004  3                 ! To collect an symbol, we keep pulling characters and
;   601          1005  3                 ! adding them to the token until we hit something that
;   602          1006  3                 ! isn't a letter or digit.  We save the final character
;   603          1007  3                 ! for later.
;   604          1008  3
;   605          1009  5                 (loop (
;   606          1010  5                         char = get_next_char();
;   607          1011  5                         if .char_class[.char] nequ tkn_k_symbol then exitloop;
;   608          1012  5                         ch$wchar(.char, .cdu$gq_token[ptr]+.cdu$gq_token[len]);
;   609          1013  5                         increment(cdu$gq_token[len]);
;   610          1014  4                 );
;   611          1015  4                 char_saved = true;
;   612          1016  4
;   613          1017  4                 ! Upcase the symbol for comparison purposes.
;   614          1018  4
;   615          1019  4                 str$upcase(cdu$gq_token,cdu$gq_token);
;   616          1020  4
;   617          1021  4                 ! Complain if the symbol is longer than 31 characters.
;   618          1022  4
;   619          1023  4                 if .cdu$gq_token[len] gtru 31 then
;   620          1024  4                         cdu$report_syntax_error(msg(cdu$_symtoolong),1,cdu$gq_token);
;   621          1025  3                 return;);
;   622          1026  3         tes;
;   623          1027  2 );
;   624          1028  2
;   625          1029  1 END;


                                                 .EXTRN  CDU$_INVCHAR, CDU$_MISSQUOTE
                                                 .EXTRN  CDU$_INTHCHAR, CDU$_SYMTOOLONG

                         0FFC 00000              .ENTRY  CDU$GET_NEXT_TOKEN, Save R2,R3,R4,R5,R6,R7,-; 0697
                                                         R8,R9,R10,R11
              0000'   CF   D5 00002              TSTL    CDU$GL_LINE_NUMBER                          ; 0881
                      0C   12 00006              BNEQ    1$                                          ;
              0000'   CF   D4 00008              CLRL    CDU$GL_CLD_ERRORS                           ; 0882
              0000'   CF   94 0000C              CLRB    RECOVERING                                  ; 0883
              0000'   CF   94 00010              CLRB    CHAR_SAVED                                  ; 0884
                      6C   95 00014 1$:          TSTB    (AP)                                        ; 0892
                      17   13 00016              BEQL    2$                                          ;
                 04   AC   D5 00018              TSTL    4(AP)                                       ;
                      12   13 0001B              BEQL    2$                                          ;
       0C        04   AC   D1 0001D              CMPL    HINT, #12                                   ; 0893
```

```
                                            0C   12 00021          BNEQ    2$
                     FF40    CF             00   FB 00023          CALLS   #0, GET_H_STRING                             ; 0894
                                   0000'    CF   B5 00028          TSTW    CDU$GQ_TOKEN                                 ; 0895
                                            01   13 0002C          BEQL    2$
                                            04   04 0002E          RET
                             08    0000'    CF   E8 0002F  2$:     BLBS    CHAR_SAVED, 3$                               ; 0909
                                          FE81   30 00034          BSBW    GET_NEXT_CHAR                                ; 0910
                     0000'   CF             50   90 00037          MOVB    R0, CHAR
                                   0000'    CF   94 0003C  3$:     CLRB    CHAR_SAVED                                   ; 0911
                     0000'   CF             01   B0 00040          MOVW    #1, CDU$GQ_TOKEN                             ; 0912
                             50    0000'    CF   9A 00045          MOVZBL  CHAR, R0                                     ; 0913
                     0000'   DF             50   90 0004A          MOVB    R0, @CDU$GQ_TOKEN+4
                     0000'   CF    0000'CF40     9A 0004F          MOVZBL  CHAR_CLASS[R0], CDU$GL_TOKEN_CLASS           ; 0918
            0F               00    0000'    CF   CF 00057          CASEL   CDU$GL_TOKEN_CLASS, #0, #15                  ; 0922
    FFD2          FFD2           FFD2     0020      0005D  4$:     .WORD   5$-4$,-
    00F1          00F1           00F1     00F1      00065                  2$-4$,-
    003E          0033           00F1     00F1      0006D                  2$-4$,-
    00F1          00F1           009D     008D      00075                  2$-4$,-
                                                                           14$-4$,-
                                                                           14$-4$,-
                                                                           14$-4$,-
                                                                           14$-4$,-
                                                                           14$-4$,-
                                                                           14$-4$,-
                                                                           6$-4$,-
                                                                           7$-4$,-
                                                                           11$-4$,-
                                                                           12$-4$,-
                                                                           14$-4$,-
                                                                           14$-4$
                                   0000'    CF   DD 0007D  5$:     PUSHL   LINE_INDEX                                   ; 0928
                                            01   DD 00081          PUSHL   #1
                                000000000G  8F   DD 00083          PUSHL   #CDU$_INVCHAR
                     0000V   CF             03   FB 00089          CALLS   #3, CDU$REPORT_SYNTAX_ERROR
                                            9F   11 0008E          BRB     2$
                     0000'   CF  000F423F   8F   D0 00090  6$:     MOVL    #999999, LINE_INDEX                          ; 0958
                                            94   11 00099          BRB     2$
                                   0000'    CF   B4 0009B  7$:     CLRW    CDU$GQ_TOKEN                                 ; 0970
                                          FE16   30 0009F  8$:     BSBW    GET_NEXT_CHAR                                ; 0972
                             53             50   90 000A2          MOVB    R0, CHAR2
                             52             53   9A 000A5          MOVZBL  CHAR2, R2                                    ; 0973
                             50    0000'CF42     9A 000A8          MOVZBL  CHAR_CLASS[R2], R0
                             03             50   91 000AE          CMPB    R0, #3                                       ; 0974
                                            0C   12 000B1          BNEQ    9$
                                000000000G  8F   DD 000B3          PUSHL   #CDU$_MISSQUOTE                              ; 0975
                     0000V   CF             01   FB 000B9          CALLS   #1, CDU$REPORT_SYNTAX_ERROR
                                            04   04 000BE          RET
                             0B             50   91 000BF  9$:     CMPB    R0, #11                                      ; 0978
                                            13   12 000C2          BNEQ    10$
                                          FDF1   30 000C4          BSBW    GET_NEXT_CHAR                                ; 0979
                     0000'   CF             50   90 000C7          MOVB    R0, CHAR
                             52             50   D1 000CC          CMPL    R0, R2
                                            06   13 000CF          BEQL    10$
                     0000'   CF             01   90 000D1          MOVB    #1, CHAR_SAVED                               ; 0983
                                            04   04 000D6          RET                                                  ; 0982
                             50    0000'    CF   3C 000D7  10$:    MOVZWL  CDU$GQ_TOKEN, R0                             ; 0988
                             50    0000'    CF   C0 000DC          ADDL2   CDU$GQ_TOKEN+4, R0
```

```
                      60          52  90 000E1          MOVB    R2, (R0)                                 ;
                           0000'  CF  B6 000E4          INCW    CDU$GQ_TOKEN                             ; 0989
                                  B5  11 000E8          BRB     8$                                       ; 0970
                      000000000G  8F  DD 000EA  11$:    PUSHL   #CDU$_INTHCHAR                           ; 0999
          00000000G   00          01  FB 000F0          CALLS   #1, LIB$SIGNAL                           ;
                                FF35  31 000F7          BRW     2$                                       ;
                                FDBB  30 000FA  12$:    BSBW    GET_NEXT_CHAR                            ; 1010
              0000'   CF          50  90 000FD          MOVB    R0, CHAR                                 ;
                      50   0000'  CF  9A 00102          MOVZBL  CHAR, R0                                 ; 1011
                      0D   0000'CF40  91 00107          CMPB    CHAR_CLASS[R0], #13                      ;
                                  13  12 0010D          BNEQ    13$                                      ;
                      51   0000'  CF  3C 0010F          MOVZWL  CDU$GQ_TOKEN, R1                         ; 1012
                      51   0000'  CF  C0 00114          ADDL2   CDU$GQ_TOKEN+4, R1                       ;
                      61          50  90 00119          MOVB    R0, (R1)                                 ;
                           0000'  CF  B6 0011C          INCW    CDU$GQ_TOKEN                             ; 1013
                                  D8  11 00120          BRB     12$                                      ; 1009
              0000'   CF          01  90 00122  13$:    MOVB    #1, CHAR_SAVED                           ; 1015
                           0000'  CF  9F 00127          PUSHAB  CDU$GQ_TOKEN                             ; 1019
                           0000'  CF  9F 0012B          PUSHAB  CDU$GQ_TOKEN                             ;
          00000000G   00          02  FB 0012F          CALLS   #2, STR$UPCASE                           ;
                      1F   0000'  CF  B1 00136          CMPW    CDU$GQ_TOKEN, #31                        ; 1023
                                  11  1B 0013B          BLEQU   14$                                      ;
                           0000'  CF  9F 0013D          PUSHAB  CDU$GQ_TOKEN                             ; 1024
                                  01  DD 00141          PUSHL   #1                                       ;
                      000000000G  8F  DD 00143          PUSHL   #CDU$_SYMTOOLONG                         ;
              0000V   CF          03  FB 00149          CALLS   #3, CDU$REPORT_SYNTAX_ERROR              ;
                                      04 0014E  14$:    RET                                              ; 1029

; Routine Size:  335 bytes,    Routine Base:  $CODE$ + 024F
```

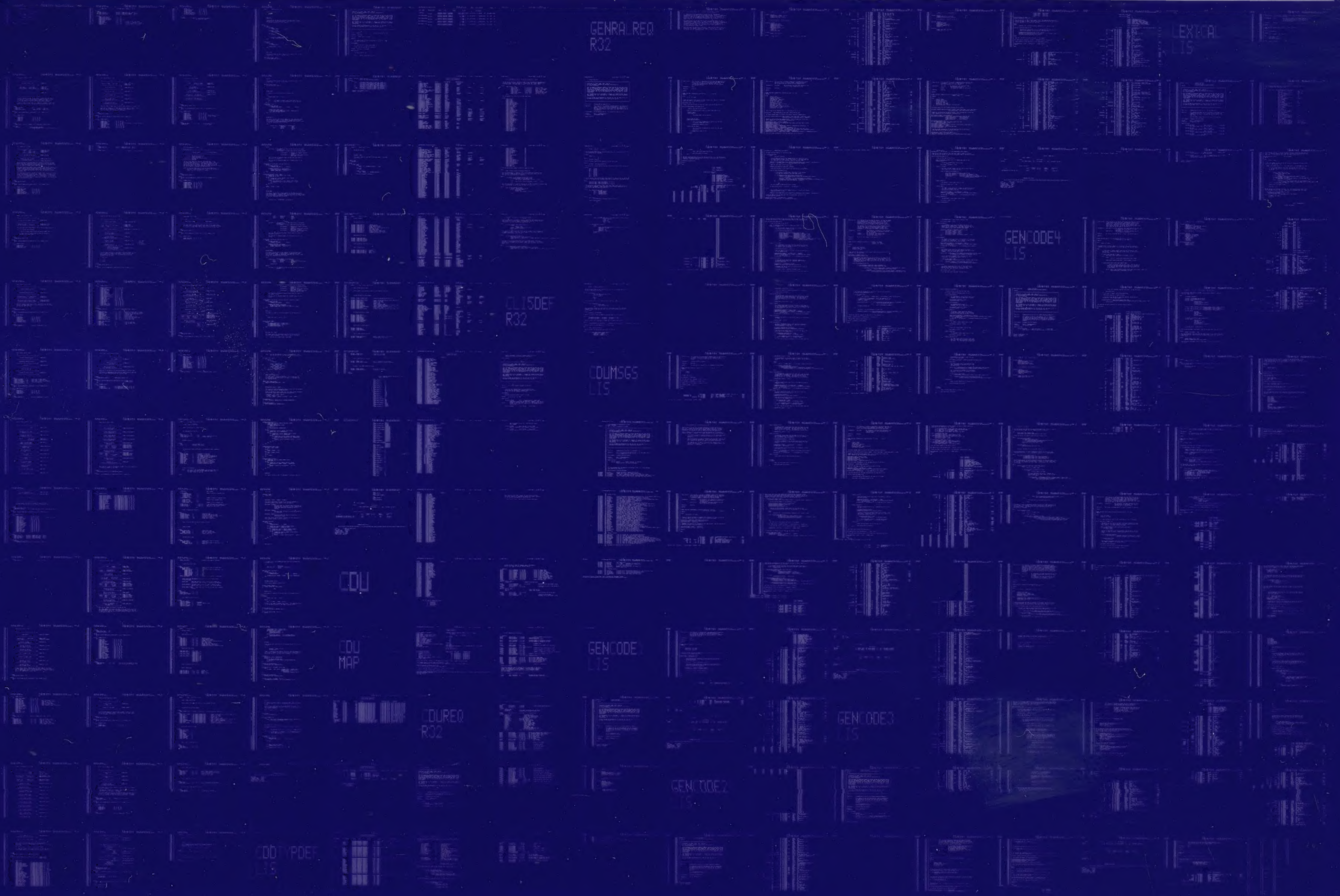
```
1 !++
1 ! Description:  This routine is called when the current token from the CLD
1 !               file must be of a specified class.  Optionally, we can also
1 !               check that the token is equal to a specified text string.
1 !
1 !               This routine also implements our simple error recovery
1 !               scheme.
1 !
1 ! Parameters:   class           By value, the required class of the token.
1 !               text_string     Optional, by reference, an ASCIC text string
1 !                               that must be equal to the token.
1 !               hint            Optional, by value, a hint to the
1 !                               CDU$GET_NEXT_TOKEN routine.  See it.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$token_must_be(class: long,
1                                  text_string: ref vector[,byte],
1                                  hint: long)                     : novalue
2 = BEGIN
2
2 builtin
2         nullparameter;
2
2 ! If we previously encountered a syntax error, then we are going to recover
2 ! from it.  Eat tokens from the CLD file until we get the one that the
2 ! caller demands be present.  Hopefully we won't encounter end of file in
2 ! the process.  If this recovery succeeds, the input token stream will
2 ! be resynchronized with the recursive descent.
2
3 if .recovering then (
3         until   .cdu$gl_token_class eqlu .class and
4                 (if nullparameter(2) then true else
4                     ch$eql(.cdu$gq_token[len],.cdu$gq_token[ptr],
4                     .text_string[0],text_string[1],%x'00'))        do (
4
4                 if token_is(tkn_k_eof) then return;
4                 cdu$get_next_token();
3         );
3         recovering = false;
2 );
2
2 ! Check that the current token is as required by the caller.  If so,
2 ! get the next token.  If not, we have a syntax error and don't get the
2 ! next token in case the required one is simply missing.
2
2 if      .cdu$gl_token_class eqlu .class and
3         (if nullparameter(2) then true else
3             ch$eql(.cdu$gq_token[len],.cdu$gq_token[ptr],
2             .text_string[0],text_string[1],%x'00'))        then
2
2         cdu$get_next_token((if nullparameter(3) then 0 else .hint))
2 else
```

```
;  684          1087  2            cdu$report_syntax_error(msg(cdu$_invitem),1,cdu$gq_token);
;  685          1088  2
;  686          1089  2  return;
;  687          1090  2
;  688          1091  1  END;


                                                                    .EXTRN  CDU$_INVITEM

                                                  000C 00000        .ENTRY  CDU$TOKEN_MUST_BE, Save R2,R3                     ; 1049
                          38        0000'    CF   E9 00002          BLBC    RECOVERING, 4$                                    ; 1064
                                             0C   11 00007          BRB     2$                                                ; 1065
                          04        0000'    CF   D1 00009  1$:     CMPL    CDU$GL_TOKEN_CLASS, #4                            ; 1070
                                             7D   13 0000E          BEQL    10$
                    FE9C  CF                 00   FB 00010          CALLS   #0, CDU$GET_NEXT_TOKEN                            ; 1071
                      04  AC        0000'    CF   D1 00015  2$:     CMPL    CDU$GL_TOKEN_CLASS, CLASS                         ; 1065
                                             EC   12 0001B          BNEQ    1$
                          02                 6C   91 0001D          CMPB    (AP), #2                                          ; 1066
                                             19   1F 00020          BLSSU   3$                                                ;
                                      08     AC   D5 00022          TSTL    8(AP)                                             ;
                                             14   13 00025          BEQL    3$                                                ;
                          50          08     AC   D0 00027          MOVL    TEXT_STRING, R0                                   ; 1068
                          51                 60   9A 0002B          MOVZBL  (R0), R1                                          ;
        51          00    0000'  DF     0000'    CF   2D 0002E          CMPC5   CDU$GQ_TOKEN, @CDU$GQ_TOKEN+4, #0, R1, -         ;
                                      01     A0      00037                  1(R0)                                             ;
                                             CE   12 00039          BNEQ    1$                                                ;
                                    0000'    CF   94 0003B  3$:     CLRB    RECOVERING                                        ; 1073
                      04  AC        0000'    CF   D1 0003F  4$:     CMPL    CDU$GL_TOKEN_CLASS, CLASS                         ; 1080
                                             35   12 00045          BNEQ    9$                                                ;
                          02                 6C   91 00047          CMPB    (AP), #2                                          ; 1081
                                             19   1F 0004A          BLSSU   5$                                                ;
                                      08     AC   D5 0004C          TSTL    8(AP)                                             ;
                                             14   13 0004F          BEQL    5$                                                ;
                          50          08     AC   D0 00051          MOVL    TEXT_STRING, R0                                   ; 1083
                          51                 60   9A 00055          MOVZBL  (R0), R1                                          ;
        51          00    0000'  DF     0000'    CF   2D 00058          CMPC5   CDU$GQ_TOKEN, @CDU$GQ_TOKEN+4, #0, R1, -         ;
                                      01     A0      00061                  1(R0)                                             ;
                                             17   12 00063          BNEQ    9$                                                ;
                          03                 6C   91 00065  5$:     CMPB    (AP), #3                                          ; 1085
                                             05   1F 00068          BLSSU   6$                                                ;
                                      0C     AC   D5 0006A          TSTL    12(AP)                                            ;
                                             04   12 0006D          BNEQ    7$                                                ;
                                             7E   D4 0006F  6$:     CLRL    -(SP)                                             ;
                                             03   11 00071          BRB     8$                                                ;
                                      0C     AC   DD 00073  7$:     PUSHL   HINT                                              ;
                    FE36  CF                 01   FB 00076  8$:     CALLS   #1, CDU$GET_NEXT_TOKEN                            ;
                                                  04 0007B          RET                                                       ;
                                    0000'    CF   9F 0007C  9$:     PUSHAB  CDU$GQ_TOKEN                                      ; 1087
                                             01   DD 00080          PUSHL   #1                                                ;
                               00000000G     8F   DD 00082          PUSHL   #CDU$_INVITEM                                     ;
                    0000V  CF                03   FB 00088          CALLS   #3, CDU$REPORT_SYNTAX_ERROR                       ;
                                                  04 0008D  10$:    RET                                                       ; 1091

; Routine Size:  142 bytes,    Routine Base:  $CODE$ + 039E
;  690        1092  1 !++
;  691        1093  1 ! Description:  This routine is called when a syntax error is encountered.
;  692        1094  1 !               It signals the error so that it will appear on the terminal.
;  693        1095  1 !               It also includes the error in the listing file, if any.
;  694        1096  1 !
;  695        1097  1 !               This routine also implements part of our simple error
;  696        1098  1 !               recovery scheme.
;  697        1099  1 !
;  698        1100  1 ! Parameters:   Standard $PUTMSG argument list.
;  699        1101  1 !
;  700        1102  1 ! Returns:      Nothing.
;  701        1103  1 !
;  702        1104  1 ! Notes:
;  703        1105  1 !--
;  704        1106  1
;  705        1107  1 GLOBAL ROUTINE cdu$report_syntax_error          : novalue
;  706        1108  2 = BEGIN
;  707        1109  2
;  708        1110  2 builtin
;  709        1111  2         argptr,
;  710        1112  2         callg;
;  711        1113  2
;  712        1114  2
;  713        1115  2 ! If we are recovering from a previous syntax error, then ignore this new
;  714        1116  2 ! one.  Doing so prevents a lot of spurious error messages.
;  715        1117  2
;  716        1118  2 if .recovering then
;  717        1119  2         return;
;  718        1120  2
;  719        1121  2 ! Signal the error along with the offending source line.
;  720        1122  2
;  721        1123  2 lib$signal(msg(cdu$_listline),nobabble+3,.cdu$gl_line_number,
;  722        1124  2                         .cld_rab[rab$w_rsz],.cld_rab[rab$l_rbf]);
;  723        1125  2 callg(argptr(),lib$signal);
;  724        1126  2
;  725        1127  2 ! Include the error in the listing file.
;  726        1128  2
;  727        1129  2 callg(argptr(),cdu$report_listing_line);
;  728        1130  2
;  729        1131  2 ! Keep track of the number of syntax errors.
;  730        1132  2
;  731        1133  2 increment(cdu$gl_cld_errors);
;  732        1134  2
;  733        1135  2 ! Set a flag saying that we are recovering from a syntax error.  This flag
;  734        1136  2 ! will be reset later when we resynchronize the input.
;  735        1137  2
;  736        1138  2 recovering = true;
;  737        1139  2 return;
;  738        1140  2
;  739        1141  1 END;



                                   0000 00000         .ENTRY  CDU$REPORT_SYNTAX_ERROR, Save nothing       ; 1107
              37      0000'  CF  E8 00002         BLBS    RECOVERING, 1$                              ; 1118
```

```
                                         0000'  CF   DD 00007           PUSHL   CLD_RAB+40                                     ; 1124
                          7E             0000'  CF   3C 0000B           MOVZWL  CLD_RAB+34, -(SP)                              ;
                                         0000'  CF   DD 00010           PUSHL   CDU$GL_LINE_NUMBER                             ; 1123
                                      00010003  8F   DD 00014           PUSHL   #65539                                         ;
                                      00000000G 8F   DD 0001A           PUSHL   #CDU$_LISTLINE                                 ;
                       00000000G  00             05  FB 00020           CALLS   #5, LIB$SIGNAL                                 ;
                       00000000G  00             6C  FA 00027           CALLG   (AP), LIB$SIGNAL                               ; 1125
                       00000000G  00             6C  FA 0002E           CALLG   (AP), CDU$REPORT_LISTING_LINE                  ; 1129
                                         0000'  CF   D6 00035           INCL    CDU$GL_CLD_ERRORS                              ; 1133
                           0000'  CF             01  90 00039           MOVB    #1, RECOVERING                                 ; 1138
                                                     04 0003E 1$:       RET                                                    ; 1141

; Routine Size:  63 bytes,    Routine Base:  $CODE$ + 042C


;  740          1142  1 END
;  741          1143  0 ELUDOM




                                                                        .EXTRN  LIB$SIGNAL

;                                 PSECT SUMMARY
;
;       Name                         Bytes                          Attributes
;
;   $OWN$                             1344  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $GLOBAL$                           276  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $PLIT$                              28  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                            1131  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                              Library Statistics
;
;                                              -------- Symbols --------        Pages        Processing
;       File                                   Total    Loaded   Percent        Mapped       Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1              18619        72         0        1000           00:01.9




;                                  COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:LEXICAL/OBJ=OBJ$:LEXICAL MSRC$:LEXICAL/UPDATE=(ENH$:LEXICAL)

; Size:            1131 code + 1648 data bytes
; Run Time:           00:25.5
; Elapsed Time:       01:06.6
; Lines/CPU Min:        2691
; Lexemes/CPU-Min:  23571
; Memory Used:  192 pages
; Compilation Complete
```

0043 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0044 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYMBOLS
LIS

NODES
LIS

OBJECT
LIS

PARSE1
LIS

PARSE3
LIS

ROUTINES
LIS

LISTING
LIS

MAIN
LIS

TABLE
LIS

PARSE2
LIS